| Ex. No: 05 | SUBQUERIES AND CORRELATED SUBQUERIES |
|---|---|
| Date | 12.09.2023 |

**Objective:**

To execute given query using the concept of subqueries and correlated subqueries

**Software Required:**

Oracle 10g

**Description:**

Subquery is usually added in the WHERE Clause of the sql statement. Most of the time, a subquery is used when you know how to search for a value using a SELECT statement, but do not know the exact value. Subqueries are an alternate way of returning data from multiple tables.

Subqueries can be used with the following sql statements along with the comparison operators like =, >, <, <=, >= etc.

- Update
- Insert
- Select
- Delete

**Correlated sub-query**

A correlated sub-query is a term used for specific types of queries in SQL in computer databases. It is a sub-query (a query nested inside another query) that uses values from the outer query in its WHERE clause. The sub-query is evaluated once for each row processed by the outer query.

**Detailed Procedure:**

Here is an example for a typical correlated sub-query. In this example we are finding the list of employees (employee number and names) having more salary than the average salary of all employees in that employee department.

SELECT employee_number, name FROM employee AS e1

WHERE salary > (SELECT avg(salary)FROM employee

WHERE department = e1.department);

In the above query the outer query is,

SELECT employee_number, name FROM employee AS e1

WHERE salary >=

And the inner query is,

(SELECT avg(salary)

FROM employee

WHERE department = e1.department);

In the above nested query the inner query has to be executed for every employee as the department will change for every row. Hence the average salary will also change. The effect of correlated sub-queries can also be obtained using outer Joins.

**Questions:**

1. List all users who have made reservations for events that are taking place in a specific venue (e.g., "USA").

```
SQL> SELECT U.Name
  2  FROM User_1128 U
  3  WHERE U.UserID IN (
  4      SELECT T.UserID
  5      FROM Ticket_1128 T
  6      INNER JOIN Event_1128 E ON T.EventID = E.EventID
  7      INNER JOIN Venue_1128 V ON E.VenueID = V.VenueID
  8      WHERE V.Country = 'USA'
  9  );

NAME
--------------------------------------------------------------------------------
John Smith
Emily Chen
Jane Doe
Sarah Adams
David Wang
Lisa Lopez
Michael Lee
Alex Kim

8 rows selected.
```

2. Find the events with the highest ticket prices.

```
SQL> SELECT E.EventID, E.Name AS EventName, MAX(T.Price) AS MaxTicketPrice
  2  FROM Event_1128 E
  3  INNER JOIN Ticket_1128 T ON E.EventID = T.EventID
  4  GROUP BY E.EventID, E.Name
  5  HAVING MAX(T.Price) = (
  6      SELECT MAX(Price) FROM Ticket_1128
  7  );

   EVENTID
----------
EVENTNAME
--------------------------------------------------------------------------------
MAXTICKETPRICE
--------------
         1
Concert in Park
           25
```

3. Find the total number of tickets reserved for a specific event.

```
SQL> SELECT E.Name AS EventName, COUNT(T.TicketID) AS TotalTicketsReserved
  2  FROM Event_1128 E
  3  LEFT JOIN Ticket_1128 T ON E.EventID = T.EventID
  4  WHERE E.Name = 'Specific Event Name'
  5  GROUP BY E.Name;

no rows selected
```

4. List the users who have made reservations with a total cost exceeding a certain amount (e.g., 50).

```
SQL> SELECT U.Name
  2  FROM User_1128 U
  3  WHERE U.UserID IN (
  4      SELECT T.UserID
  5      FROM Ticket_1128 T
  6      GROUP BY T.UserID
  7      HAVING SUM(Price) > 50
  8  );

no rows selected
```

5. Retrieve the events where the number of reservations exceeds a certain threshold.

```
SQL> SELECT E.Name AS EventName, COUNT(T.TicketID) AS TotalReservations
  2  FROM Event_1128 E
  3  LEFT JOIN Ticket_1128 T ON E.EventID = T.EventID
  4  GROUP BY E.Name
  5  HAVING COUNT(T.TicketID) > 10;

no rows selected
```

6. Find all users who have made more reservations than the average number of reservations across all users.

```
SQL> SELECT U.Name
  2  FROM User_1128 U
  3  WHERE U.UserID IN (
  4      SELECT T.UserID
  5      FROM Ticket_1128 T
  6      GROUP BY T.UserID
  7      HAVING COUNT(T.TicketID) > (
  8          SELECT AVG(ReservationCount)
  9          FROM (
 10              SELECT COUNT(T2.TicketID) AS ReservationCount
 11              FROM Ticket_1128 T2
 12              GROUP BY T2.UserID
 13          )
 14      )
 15  );

no rows selected
```

7. List all events where the total ticket price of reservations exceeds a certain amount.

```
SQL> SELECT E.Name AS EventName
  2  FROM Event_1128 E
  3  WHERE (
  4      SELECT SUM(T2.Price)
  5      FROM Ticket_1128 T2
  6      WHERE T2.EventID = E.EventID
  7  ) > 100;

no rows selected
```

8. Find the users who have made reservations for more than one event.

```
SQL> SELECT U.Name
  2  FROM User_1128 U
  3  WHERE (
  4      SELECT COUNT(DISTINCT T2.EventID)
  5      FROM Ticket_1128 T2
  6      WHERE T2.UserID = U.UserID
  7  ) > 1;

no rows selected
```

9. Retrieve the events with the highest number of reservations.

```
SQL> SELECT E.Name AS EventName
  2  FROM Event_1128 E
  3  WHERE (
  4      SELECT COUNT(T2.TicketID)
  5      FROM Ticket_1128 T2
  6      WHERE T2.EventID = E.EventID
  7  ) = (
  8      SELECT MAX(ReservationCount)
  9      FROM (
 10          SELECT COUNT(T3.TicketID) AS ReservationCount
 11          FROM Ticket_1128 T3
 12          GROUP BY T3.EventID
 13      )
 14  );

EVENTNAME
----------------------------------------------------------------
Concert in Park
Movie Night
Sports Tournament
Art Exhibition
```

10. For each event, find the number of reservations made by users.

```
SQL> SELECT E.Name AS EventName, COUNT(T.TicketID) AS TotalReservations
  2  FROM Event_1128 E
  3  LEFT JOIN Ticket_1128 T ON E.EventID = T.EventID
  4  GROUP BY E.Name;

EVENTNAME
--------------------------------------------------------------------------------
TOTALRESERVATIONS
-----------------
Concert in Park
                2

Sports Tournament
                2

Movie Night
                2


EVENTNAME
--------------------------------------------------------------------------------
TOTALRESERVATIONS
-----------------
Food Festival
                0

Dance Workshop
                0

Comedy Show
                0


EVENTNAME
--------------------------------------------------------------------------------
TOTALRESERVATIONS
-----------------
Art Exhibition
                2

Tech Conference
                0


8 rows selected.
```

11. Find the events for which the total ticket price of reservations exceeds the average total ticket price for all events.

```
SQL> SELECT E.Name AS EventName
  2  FROM Event_1128 E
  3  GROUP BY E.Name
  4  HAVING SUM(
  5      (SELECT SUM(T2.Price)
  6       FROM Ticket_1128 T2
  7       WHERE T2.EventID = E.EventID)
  8  ) > (
  9      SELECT AVG(TotalPrice)
 10      FROM (
 11          SELECT SUM(T3.Price) AS TotalPrice
 12          FROM Ticket_1128 T3
 13          GROUP BY T3.EventID
 14      )
 15  );

EVENTNAME
----------------------------------------------------------
Concert in Park
Movie Night
```

12. List users who have made reservations for multiple events on the same day.

```
SQL> SELECT U.Name
  2  FROM User_1128 U
  3  WHERE U.UserID IN (
  4      SELECT T.UserID
  5      FROM Ticket_1128 T
  6      INNER JOIN Event_1128 E ON T.EventID = E.EventID
  7      GROUP BY T.UserID, E.EventDate
  8      HAVING COUNT(DISTINCT E.EventDate) > 1
  9  );

no rows selected
```

**Result:**

The given queries were executed successfully using set operators and joins.